Journal of Network and Systems Management (2024) 32:59
https://doi.org/10.1007/s10922-024-09831-x

# Benchmarking Large Language Models for Log Analysis, Security, and Interpretation

Egil Karlsen[1] · Xiao Luo[2] · Nur Zincir-Heywood[1] · Malcolm Heywood[1]

Received: 24 November 2023 / Revised: 24 November 2023 / Accepted: 10 May 2024 /
Published online: 13 June 2024

## Abstract

Large Language Models (LLM) continue to demonstrate their utility in a variety of emergent capabilities in different fields. An area that could benefit from effective language understanding in cybersecurity is the analysis of log files. This work explores LLMs with different architectures (BERT, RoBERTa, DistilRoBERTa, GPT-2, and GPT-Neo) that are benchmarked for their capacity to better analyze application and system log files for security. Specifically, 60 fine-tuned language models for log analysis are deployed and benchmarked. The resulting models demonstrate that they can be used to perform log analysis effectively with fine-tuning being particularly important for appropriate domain adaptation to specific log types. The best-performing fine-tuned sequence classification model (DistilRoBERTa) outperforms the current state-of-the-art; with an average F1-Score of 0.998 across six datasets from both web application and system log sources. To achieve this, we propose and implement a new experimentation pipeline (LLM4Sec) which leverages LLMs for log analysis experimentation, evaluation, and analysis.

**Keywords** Log analysis · LLM · GPT · BERT · NLP · Security · Interpretation

✉ Egil Karlsen
egil.karlsen@dal.ca

Xiao Luo
xiao.luo@okstate.edu

Nur Zincir-Heywood
zincir@cs.dal.ca

Malcolm Heywood
mheywood@cs.dal.ca

[1] Faculty of Computer Science, Dalhousie University, University Ave., Halifax, NS B3H 1W5, Canada

[2] Department of Management Science and Information Systems, Oklahoma State University, 370 Business Building, Stillwater, OK 74078, USA

## 1 Introduction

In today's digital landscape, networks, especially within enterprise infrastructure, are extensive and active 24/7, often catering to hundreds of thousands of users concurrently. This continuous activity implies that log files are generated at the same rate by applications, services, systems, and networks; where log files are integral to cybersecurity monitoring, forensics, and alert mechanisms. With high user traffic, these logs can comprise hundreds of millions of records from diverse architectural layers. Traditionally, these voluminous and varied logs are aggregated via centralized systems like SIEMs, facilitating log analysis and anomaly detection.

Conventional log analysis relies on static regular expression parsers (e.g. DRAIN [1]) to extract salient features from the aggregated logs. However, this methodology poses significant constraints, primarily because parsers must be preconfigured for specific log formats, and given precise instructions on the useful features that should be extracted. This is where Natural Language Processing (NLP) techniques offer substantial advantages. NLP enables dynamic feature extraction without the prerequisite of prior knowledge about data structures, enhancing scalability and, crucially, versatility [2, 3].

Existing research underscores the potential of NLP in application, network, service, and system log analysis. Studies indicate that semantic information from word2vec models, coupled with syntactic insights from TF–IDF, can yield meaningful vector space representations [3]. Moreover, Karlsen et al. have demonstrated the advantage of Large Language Models (LLMs)-based semantic feature extraction over traditional syntactic methods in analyzing application logs for security analysis [4]. Specifically, their research highlights the efficacy of employing DistilRoBERTa, a prominent LLM, for generating sentence embeddings from raw log data, by evaluating its ability to disambiguate abnormal events using unsupervised machine learning methods.

Despite these advancements, the exploration of NLP-based log analysis, especially through LLMs, has room to improve. This study aims to bridge the gap by delving into an LLM-based approach for feature extraction in log analysis. It benchmarks various LLMs across application, system, and network-level log datasets, evaluating the approach's versatility for understanding anomalous behaviour. Furthermore, this study investigates the benefits of domain adaptation via the fine-tuning of LLMs.

Building on the earlier work of Karlsen et al., this research employs five distinct LLM architectures, evaluated using six labelled log datasets for security. It also evaluates the advantages of domain adaptation through the individual fine-tuning of LLMs using sequence classification, a supervised classification method. The contributions of this research are as follows:

- *Evaluate Five LLMs* assess the efficacy of five distinct Large Language Models in identifying abnormal events and analyzing application and system logs, thereby establishing a performance benchmark in the field.

- *Demonstrate Versatility* confirm the adaptability and broad applicability of LLM-based approaches through tests across six log files.
- *Optimize with Fine-Tuning* explore the impact of advanced fine-tuning on LLMs using sequence classification, to enhance log analysis precision.
- *Boost Interpretability* employ visualizations and case studies to deepen the understanding and interpretation of the LLM-based approach in log analysis.
- *Provide Experimental pipeline* develop "LLM4Sec," a pipeline for experimenting with LLM-based log analysis for security.

The remaining sections of this paper are organized as follows: Sect. 2 reviews the literature in the field. Section 3 introduces the proposed approach and discusses the datasets, feature extraction, domain adaptation, and supervised learning models used in this research. Section 4 presents the evaluations and the results obtained. Finally, conclusions are drawn and future work is discussed in Sect. 5.

## 2 Literature Review

In this section, we give an overview of previous work on log analysis via NLP and LLM-based approaches.

### 2.1 Traditional NLP Approaches

Nguyen and Franke [5] propose a supervised ensemble learning system that combines outputs from various supervised machine learning models to optimize abnormal security event classification. They test this pipeline on the ECML/PKDD 2007 and HTTP CSIC 2010 datasets using hand-picked features and log line syntactic characteristics. They emphasize the importance of adapting intrusion detection systems (IDSs) to heterogeneous and adversarial network environments. The proposed IDS outperforms baseline supervised machine learning methods by 10% in accuracy, achieved through an online learning pipeline. The IDS incorporates supervised machine learning algorithms, including decision stump, Naive Bayes, RBF Network, and Bayesian Network. A significant contribution is a novel pipeline for fusing IDS outputs to enhance accuracy and reliability. The IDS achieves an accuracy of 0.9098 in the CSIC dataset and 0.9256 in the PKDD/ECML dataset, albeit with higher computational costs for model output fusion.

Moradi Vartouni et al. [6] explore novel feature extraction techniques for intrusion detection using deep belief networks and parallel feature fusion methods. They apply synthetic feature extraction and feature fusion to bi-gram representations of log files, alongside dimensionality reduction algorithms like FICA, PCA, and KPCA. Their evaluation accesses the impacts of performing dimensionality reduction, synthetic feature fusion, and baseline n-gram representations for training unsupervised anomaly detection algorithms (Isolation forest, Elliptic envelope, and one-class SVM). Benchmarking is conducted using the ECML/PKDD 2007 and HTTP CSIC datasets. Results suggest that deep learning neural networks, especially fusion models, provide significant advantages in feature extraction for anomaly detection. Notably, the feature fusion method achieves an accuracy of 0.8535 for Isolation forest in the CSIC dataset and 0.8493 for Elliptic envelope in the PKDD dataset after extensive exploration and parameter tuning.

Bhatnagar et al. [7] benchmark the performance of four supervised machine learning algorithms for intrusion detection. Their solution extracts features from raw log files using classical neural networks and deep belief networks. The top five highest entropy features are selected from the ECML/PKDD 2007 and HTTP CSIC 2010 datasets and a bag-of-words vector space complements these features. This hybrid feature set trains four supervised classifiers (MLP classifier, Multinomial classifier, decision tree, and random forest), all achieving an average accuracy of 0.996 in identifying attacks across both datasets.

Farzad and Gulliver [8] propose a supervised approach to anomaly detection which explores the use of word frequency in security log characteristics in combination with autoencoders and GRU, LSTM, and Bi-LSTM for anomaly detection. There are two auto-encoders in their proposed method, individually trained on negative (anomalous) and positively (normal) labelled log word frequency data with the aim of extracting meaningful features toward either class. The success of the autoencoders in extracting meaningful representations of the preprocessed log data is evaluated using deep learning methods GRU, LSTM, and Bi-LSTM. They name the different classification methods Auto-GRU, Auto-LSTM, and AutoBLSTM respectively, and evaluate on publicly available system log files.

Tümer Sivri et al. [9] experimented with a variety of different machine learning approaches applied to character features from the ECML/PKDD 2007 and HTTP CSIC 2010 datasets. They pre-process the datasets by extracting handpicked log features and converting these features to lowercase UTF-8 encodings. They identify that the LSTM deep learning approach has the best classification performance in intrusion detection applied to the character-level representations. This work requires careful selection and extraction of features which makes the approach difficult to apply to other log formats.

Adhikari and Bal [10] experiment with carefully selected log features from the HTTP CSIC 2010 dataset which range from statistical log line characteristics like character count to request type. They explore these feature representations and evaluate their ability to aid in intrusion detection when combined with several machine learning techniques, where XGBoost achieves the best results. This methodology also relies on carefully selected features and as such also suffers from limited adaptability for different log formats.

Copstein et al. [11] investigate syntactic features based on the TF–IDF technique for intrusion detection in application log files. They use TF–IDF to rank terms based on uniqueness in the entire dataset, suggesting this feature set as a means to extract the form of log line data and syntactical attributes crucial for anomaly detection. Unsupervised clustering algorithms (K-Means, DBScan, and EM clustering) evaluate this approach using three datasets: ISOT-CID, ECML/PKDD 2007, and an Indonesian Apache Access datasets. The syntactic approach outperforms traditional log abstraction techniques, consistently disambiguating malicious behaviour in various traffic log data forms. The highest performance is achieved with K-Means clustering, with accuracy scores ranging from 0.6 to 0.7 for anomaly detection across the three datasets.

Expanding on the previously discussed work, Karlsen et al. [4] continue the exploration of syntactic feature extraction with application log files and compare it against a semantic approach. The semantic approach utilizes an LLM-based feature extraction method which extracts a sentence embedding representation from the raw logs. They utilize the DistilRoBERTa model and perform a comparison of this new method against the short syntactic methodology outlined by Copstein et al. [11] using four unsupervised machine learning algorithms on three datasets, ECML/PKDD 2007, CSIC 2010, and the Indonesian Apache Access datasets. The resulting study notes that the semantic approach outperforms that of the short syntactic method by about 5% and has the most potential for further development.

### 2.2 LLM-Based Approaches

In recent work, Nam et al. [12] explore BERT for log analysis in Virtual Machine (VM) failure prediction. They propose a model where BERT extracts sentence embeddings for each log, which are then fed into a pre-trained convolutional neural network for predicting failures within a 2 to 30-min window. They achieve an accuracy of 0.74 on an in-house OpenStack VM system log test and training set, which is not publicly available. A notable benefit of this BERT–CNN model is that the embedding representation enables the prediction of VM failure even in cases not observed during training.

Wang et al. [13] propose an intrusion detection system that utilizes word2vec and TF–IDF as a feature extraction method with supervised methods[1] and is benchmarked for classification. This system is evaluated on the supercomputer system log datasets with the best results indicating that TF–IDF/word2vec extraction methods and LSTM perform with an accuracy of 0.99.

Research by Qi et al. [14] proposes another deep learning-based anomaly detection system called Adanomaly which performs anomaly detection using a combination of log key extraction using the Drain log parser [1], and an adversarial autoencoder method for reconstructive error-based anomaly detection. They evaluate this model using three system log files with an accuracy changing between 0.92 and 0.98. However, the use of the log parser, Drain, to extract log key representations reduces the generalizability of the approach.

Recently, Seyyar et al. [15] proposed an intrusion detection system specifically for web applications that first extracts the URL from the application logs, and then generates an embedding using an NLP transformer model—BERT. This provides the embedding for a CNN classifier for performing intrusion detection. Inference speeds of 0.4 ms are reported at an average accuracy of 0.96 when evaluated on the CSIC, FWAF, and HTTPparams datasets. However, this system performs URL feature extraction as part of the pre-processing step, a process that limits its generalization.

[1] Gradient boosted decision tree, Naive Bayes classifier, and LSTM.

Guo et al. [16] propose an anomaly detection system called LogBERT. LogBERT leverages the masked language modelling feature of the NLP transformer model BERT to perform a self-supervised form of anomaly detection on pre-parsed logs extracted using Drain. The model is designed to identify key features in the log representation, thus making predictions about what feature will appear next. The LogBERT system is applied to system logs such as BGL and Thunderbird system datasets and the model reports accuracies of 0.8232, 0.9083, and 0.9664 respectively. Notably, this method also relies on the rule-based log parsing methodology of Drain to extract meaningful log key representations, i.e. limited to scenarios for which appropriate rules exist.

Shao et al. [17] propose an improvement on the LogBERT model called Prog-BERT–LSTM. This builds on LogBERT by adapting the sequence feature learning components with an LSTM. The semantic understanding of log sequence is improved, and as a result, when applied to the BGL and Thunderbird systems datasets they are able to report accuracies from approximately 0.84 to 0.97. Notably, this method also relies on the Drain rule-based log parsing methodology to extract meaningful log key representations and hence inherits the same limitations.

A study by Guo et al. [18] proposes a novel supervised transformer-based pipeline for log anomaly detection called TransLog. The proposed system leverages the enhanced sentence embeddings of the sentence-BERT LLM to vectorize pre-processed log data in the form of log templates extracted using Drain. The embeddings of the log templates are used to pre-train a transformer encoder model in which a classification head is employed for the task of anomaly detection. They add a 'light adapter' to the transformer architecture to enable faster fine-tuning through targeted training of specific components of the pre-trained model. This approach is evaluated using publicly available system logs such BGL, and Thunderbird achieving F1-scores around 0.99. Naturally, this approach is reliant on pre-processed system logs due to the use of Drain to extract log template representations and so relies on a static log parser.

Le and Zhang [19] attempt to incorporate the BERT language model as an approach to vectorizing pre-processed log lines. This representation is then fed into a transformer encoder block which alongside a softmax layer classifies anomalous events. The resulting model named NeuralLog is evaluated on the BGL, Thunderbird, HDFS, and Spirit datasets. This system aims to replace the out-of-vocabulary limitations of typical log parsing approaches that struggle to adapt to new words in the log file, i.e. Out Of Vocabulary words (OOV). The resulting model provides performance around an F1-score of 0.96. It should be noted that this approach still requires pre-processing of the log files in order to encode and classify.

Sec2vec is an NLP transformer-based method for web log vectorization proposed by Gniewkowski et al. [20] in 2023. This system uses the RoBERTa language model to perform log feature extraction, through fine-tuning of both the original model and its tokenizer. Additionally, the tokenizer has had its tokenization strategy swapped for a Byte-level Byte Pair Encoder in order to benefit from improved token-level representations with Byte-level encoding representations of the underlying text. As part of the process for preparation, this system is fine-tuned using the log file. Afterwards, an embedding representation can be extracted from the same log file through

**Table 1** F1-score results from literature

| Study | CSIC | PKDD | Thunderbird | BGL | Spirit |
|---|---|---|---|---|---|
| Nguyen and Franke [5][a] | 0.9098 | 0.9256 | – | – | – |
| Bhatnagar et al. [7][a] | 0.996 | 0.996 | – | – | – |
| Wang et al. [13] | – | – | 0.99 | – | – |
| Seyyar et al. [15] | 0.96 | – | – | – | – |
| Gniewkowski et al. [20] | 0.98 | – | – | – | – |
| Farzad and Gulliver [8][a] | – | – | 0.993 | 0.994 | – |
| Guo et al. [18] | – | – | 0.998 | 0.98 | – |
| Le and Zhang [19] | – | – | 0.96 | 0.98 | 0.97 |
| Tümer Sivri et al. [9] | 0.982 | – | – | – | – |
| Adhikari and Bal [10][a] | 0.93 | – | – | – | – |

[a] Indicates a result in which cited work reports Accuracy rather than F1-score

the concatenation of the hidden states from the final four layers of the LLM. This embedding representation of the log is evaluated using the Random Forest classifier. Benchmarking is performed using web service, and URL datasets specifying the vector size as 3072. Sec2Vec returns F1-scores (in average) around 0.98. It should be noted that an embedding space of 3072 is verbose and while the study does not report computational costs for any of the steps, fine-tuning, training, and performing inference with this vector space will be expensive. Additionally, they do not perform an ablation study, so it is not clear how much the computationally intensive steps contribute to the performance.

## 2.3 Summary

The literature presents several gaps in NLP-based log analysis, as follows.

- Log analysis research typically explores either web application logs with the publicly available CSIC, and PKDD datasets, or they evaluate performance on publicly available system logs such as Thunderbird, and BGL (Table 1). In practice, systems will be deployed under both settings, so should be benchmarked on both types of data.
- The LLMs employed focus on a limited set of LLM architectures, Table 2. However, the literature for state-of-the-art transformer language models provides several alternatives to the BERT/RoBERTa models that have unique differences or improvements that have yet to be explored in log analysis.
- Performance is only assessed in terms of the modified LLM. As such it is difficult to determine how much performance improvement results from the 'optimization' versus the original LLM.

Thus, in this paper, our goal is to explore the versatility of LLM-based models for log analysis with several language model architectures and evaluate them using the

**Table 2** Overview of supervised methodologies used in previous LLM studies, RegEx indicates that useful features were handpicked and extracted from the logs

| Study | Pre-parsed | Traditional NLP | LLM-based |
|---|---|---|---|
| Nguyen and Franke [5] | – | YES | NO |
| Bhatnagar et al. [7] | – | YES | NO |
| Farzad and Gulliver [8] | – | YES | NO |
| Wang et al. [13] | – | YES | NO |
| Tümer Sivri et al. [9] | RegEx | YES | NO |
| Adhikari and Bal [10] | RegEx | YES | NO |
| Seyyar et al. [15] | RegEx | NO | BERT |
| Seyyar et al. [15] | URL extraction | NO | BERT |
| Gniewkowski et al. [20] | – | NO | RoBERTa |
| Guo et al. [18] | Drain | NO | BERT |
| Le and Zhang [19] | Drain | NO | BERT |

original language models as well as the fine-tuned language models to understand the effects (if any) and requirements of the fine-tuning process. The results of this study provide a benchmark and foundation for LLM-based feature extraction in log analysis for intrusion detection.

## 3 Methodology

In the following, we first summarize the various properties of the datasets adopted for benchmarking (Sect. 3.1). Section 3.2 introduces the LLM4Sec pipeline[2] as a workflow applied to log files encompassing a portfolio of LLM (Sect. 3.3), visualization routines (Sect. 3.5) and performance metrics. Such a workflow is generic in that no additional pre- or post-processing steps are deployed.

### 3.1 Datasets

This study explores the efficacy of a general approach to log analysis which leverages the flexibility of feature engineering approaches in semantic analysis on six log files, three are web application logs, and three are system logs. Since a primary component of this research is to explore a general approach to application and system log analysis for security purposes, it is important to employ data from different sources. Therefore, six publicly available datasets are employed in this research. Each dataset represents a different real-life system as summarized below, in Tables 3 and 4. For the purposes of this research, the labels were reduced to a binary characterization of the task. Thus, classes associated with malicious behaviour were all

[2] LLM4Sec codebase is available at https://github.com/EgilATKarlsen/LLM4Sec

**Table 3** Application logs

| Class | AA | | CSIC | | PKDD | |
|---|---|---|---|---|---|---|
| | # | % | # | % | # | % |
| Normal | 29,778 | 85.71 | 36,000 | 59.01 | 35,006 | 69.85 |
| Anomalous | 4965 | 14.29 | 25,000 | 30.99 | 15,109 | 30.15 |
| Total | 34,743 | 100 | 61,000 | 100 | 50,115 | 100 |

**Table 4** System log properties

| Class | Thunderbird | | Spirit | | BGL | |
|---|---|---|---|---|---|---|
| | # | % | # | % | # | % |
| Normal | 207,963,953 | 98.46 | 90,656,272 | 33.29 | 4,399,503 | 92.66 |
| Anomalous | 3,248,239 | 1.54 | 181,642,697 | 66.71 | 348,460 | 7.34 |
| Total | 211,212,192 | 100 | 272,298,969 | 100 | 4,747,963 | 100 |

labelled as ANOMALOUS and safe activity was labelled as NORMAL. The details of each dataset are described as follows:

### 3.1.1 Application Logs

Application logs, combining service-specific log files and often network traffic, provide a comprehensive view of application and network-level interactions for security-related events. The Apache Access Dataset, published in 2022 by Indramayu State Polytechnic University, focuses on intrusion detection in web server access logs, including 37,693 records of normal, suspicious, and attack queries, representing typical non-anonymized application logs. Hereafter, this dataset is referred to as AA [21]. The CSIC Dataset, introduced by the Information Security Institute of the Spanish Research National Council in 2010, serves as an updated resource for intrusion detection research in web application logs. The dataset simulates an e-commerce environment with various attack types like SQL injection and buffer overflow and is notable for its non-anonymized data [22]. Finally, the ECML/PKDD Dataset from the 2007 Discovery Challenge, referred to as PKDD, comprises 50,115 log records with HTTP packet details for anomaly detection, including a wide range of attack behaviours like XSS and SQLi, characterized by its anonymized data for privacy, making it the most verbose dataset in terms of log line representation [23].

### 3.1.2 System Logs

A set of datasets was published in 2007 by Oliner and Stearley [24] that consists of a series of labelled system/event logs produced by supercomputer clusters in June 2006. Events are labelled with alert and non-alert tags. Each dataset is named after the supercomputer from which they originated and the three datasets used in this work are Thunderbird, BGL, and Spirit. These logs differ significantly from the

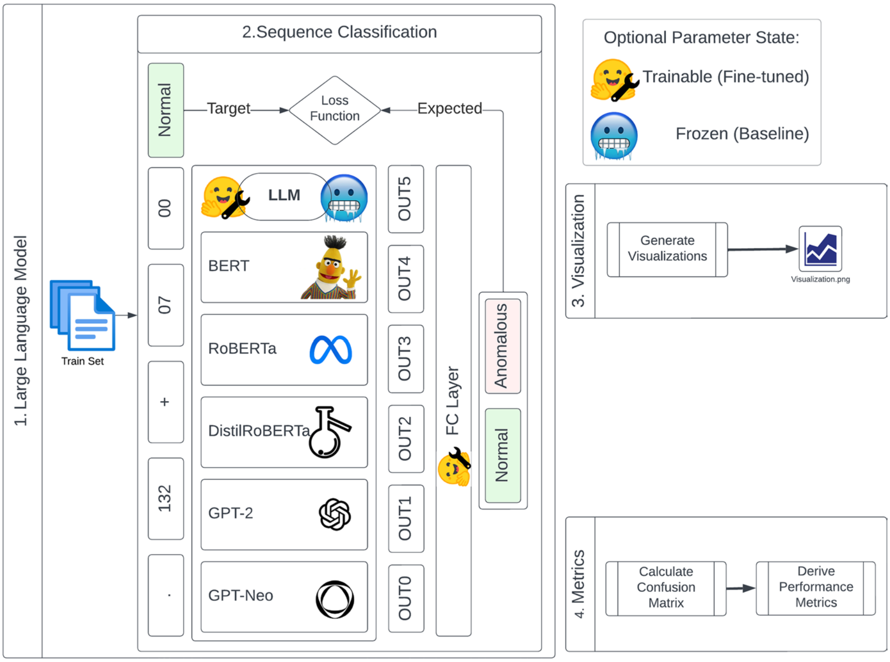


**Fig. 1** LLM4Sec pipeline overview. The original LLM architecture is augmented with a fully connected (FC) layer for the purpose of anomaly detection

previous application-style logs in that they are a record of processor-level activity, and are significantly more volumous as seen in Table 4.

### 3.2 LLM4Sec Pipeline

As part of this research, a new benchmarking experimentation pipeline is developed in order to streamline the research conducted in this study. LLM4Sec is a flexible and componentized CLI tool for log analysis that enables researchers to easily evaluate the performance of different LLMs for the task of log analysis. Integration of methods for interpretation and visualization, and fine-tuning options provide the necessary utilities to expand and continue this research. LLM4Sec deploys the same sequence of ‘modules’ that the experimental structure of this study follows: fine-tuning, visualizations, and performance analysis, Fig. 1. The LLM-related elements assume the Huggingface platform,[3] supporting a wide range of models. In addition to the LLM-based feature extraction methods, syntactic approaches such as TF–IDF, and the statistical NLP approach provided by Copstein et al. are also available for experimentation [11].

[3] https://huggingface.co.

### 3.3 Large Language Models

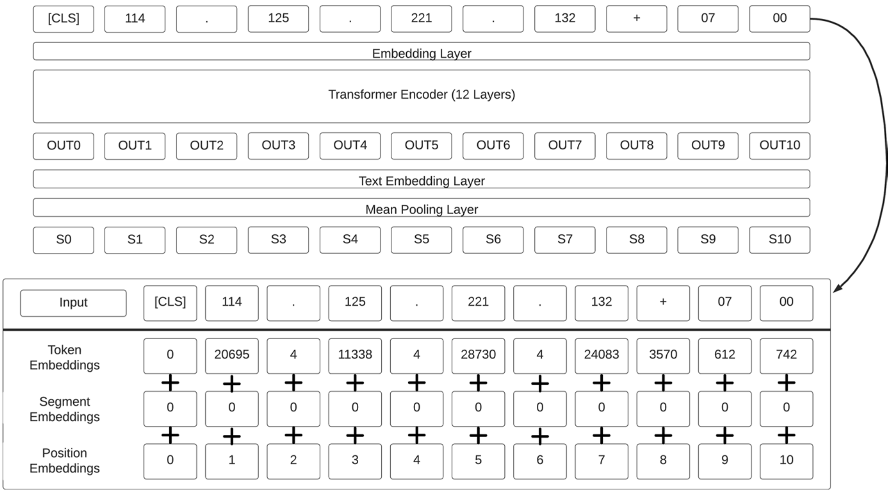
This module of the LLM4Sec pipeline, shown in Fig. 1, utilizes the log data to fine-tune the pre-trained LLMs and generate vectorized log representations that embed the semantic relations between the tokens in the log. As the first module in the pipeline, it accepts the raw log file as input, then divides the log file into 70% training, 10% validation and 20% testing, and provides log-specific LLMs and a vectorized log representation as the output feeding into the subsequent module. The details of the LLMs evaluated in this research are provided in the following subsections.

#### 3.3.1 BERT

The BERT model (Bidirectional Encoder Representations from Transformers) revolutionized NLP upon its release by Google in 2018, employing a bidirectional approach to the interpretation of textual data based on the transformer architecture introduced by Vaswani et al. [25]. Trained on an expansive corpus of 16 GB, including over 3.3 billion words from Wikipedia and the Brown Corpus, BERT's architecture of 12 transformer encoder layers, each with 768 hidden states, leverages multi-head self-attention and feed-forward layers for contextual learning of words in a sentence. Tokenization is managed by the WordPiece algorithm [26], which breaks down words into recognizable subunits for the model. BERT's training utilizes masked language modelling-masking 15% of words for context-based prediction-and next-sentence prediction, enhancing its sequential understanding. These techniques enable fine-tuning for specific domains, as shown in the LLM4Sec fine-tuning module in Fig. 1. In order to generate meaningful embedding representations of logs we employ sentence embedding vectorization. This is accomplished by performing mean pooling on the final layer of the LLM to reduce the logits to a 768-dimension vector. The process is outlined in the model's architectural visualization for sentence embedding extraction in Fig. 2.

#### 3.3.2 RoBERTa

The RoBERTa model, developed by Facebook AI in 2019, improves on the original BERT architecture with several key additions, earning its name as the Robustly Optimized BERT Approach [27]. It was trained on a massive 160 GB text corpus, extending beyond BERT's 16 GB dataset with additional web and news content. Notably, RoBERTa omits the Next-sentence Prediction (NSP) task, a decision informed by evidence that NSP was not crucial to BERT's success. RoBERTa also adopts a different pre-training configuration, utilizing larger batch sizes and fewer steps to achieve lower perplexity and quicker training. Unlike BERT's static approach, RoBERTa introduces dynamic masking during training, where the masked tokens vary during the learning process. Lastly, it uses Byte Pair Encoding (BPE) for tokenization, which results in a more extensive vocabulary of over 50,000 tokens compared to BERT's 30,000, facilitating a more nuanced understanding of language. This LLM provided improved performance over the BERT model in many



| Input | [CLS] | 114 | . | 125 | . | 221 | . | 132 | + | 07 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Token Embeddings | 0 | 20695 | 4 | 11338 | 4 | 28730 | 4 | 24083 | 3570 | 612 | 742 |
| | + | + | + | + | + | + | + | + | + | + | + |
| Segment Embeddings | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | + | + | + | + | + | + | + | + | + | + | + |
| Position Embeddings | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Fig. 2** This figure shows the sentence embedding extraction technique applied to a CSIC log line using BERT

areas of NLP, motivating its inclusion in this evaluation of LLM performance in log analysis.

### 3.3.3 DistilRoBERTa

In 2019, Sanh et al. introduced the concept of distillation in NLP, a method to streamline large BERT-style models for efficiency, as seen in their study on DistilRoBERTa [28]. This technique uses a teacher-student model where a smaller "student" model learns to replicate the larger "teacher" model's outputs. The student's training focuses on minimizing the difference between its outputs and the teacher's, effectively transferring knowledge. Applied to models like BERT and RoBERTa, distillation reduces the layers, parameters, and training time significantly, resulting in faster inference times. DistilRoBERTa, for instance, uses 6 layers and 82 million parameters, compared to RoBERTa's 12 layers and 125 million parameters, halving the inference time while maintaining output fidelity. This model was selected as the smallest of the LLMs being evaluated, it aims to provide the robust performance of RoBERTa in half the computation cost, which is particularly important in the case of log analysis where inference time is a significant factor in the setting of real-time intrusion detection.

### 3.3.4 GPT-2

OpenAI's GPT-2, released in 2019, is a departure from BERT-style models, with its unique autoregressive, decoder-only transformer architecture designed for translation, text generation, and question answering [29]. It is uni-directional, masking future tokens during training to prevent the model from using future context,

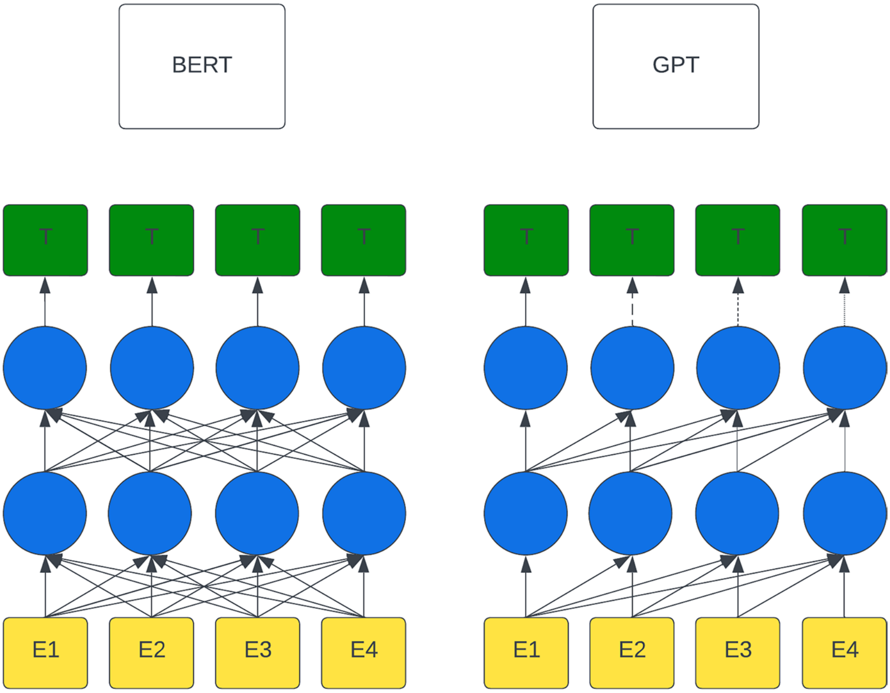


**Fig. 3** A simplified visualization of how the encoder-based architecture of BERT differs from the decoder-based architecture of GPT

unlike the bi-directional approach of BERT. The model uses self-attention, encoder-decoder attention, and feed-forward layers for contextual processing and next-token prediction. Trained on a diverse 40 GB dataset from the web, GPT-2's full version has 1.5 billion parameters, though a smaller variant with 124 million parameters was used for this research [29]. It shares the BPE tokenization method with RoBERTa, having a similar-sized vocabulary of over 50,000 tokens. To facilitate sentence embedding extraction like BERT, padding tokens are introduced to GPT-2, aligning it with BERT's method for fixed-length sentence embeddings by mapping a padding token to the existing eos_token in the tokenizer. This LLM was selected to provide a contrasting perspective to that of the encoder-only transformer LLMs, providing log analysis results from a completely different architecture (Fig. 3).

### 3.3.5 GPT-Neo

The GPT-Neo model is an Eleuther AI replication of the closed source GPT-3 model produced by Open AI and as such is an autoregressive decoder style language model [30]. This model also assumes a casual language model and utilizes a slightly modified autoregressive decoder-style architecture to that of GPT-2. GPT-Neo differs from GPT-2 in that it utilizes local attention between every other layer, and was trained on an 825 GB dataset or 'the pile'. Training using the pile exposes the LLM

**Table 5** Model characteristic of the different large language models employed

| Language model | Word embedding space | Parameters | Training data size (GB) |
|---|---|---|---|
| BERT-base-cased | 768 | 110M | 16 |
| RoBERTa | 768 | 123M | 160 |
| DistilRoBERTa-base | 768 | 82M | 40 |
| GPT-2 | 768 | 124M | 40 |
| GPT-Neo | 2048 | 1.3B | 800 |

to a wide variety of different text sources including books, GitHub repositories, webpages, chat logs, as well as research papers from a number of STEM fields. The GPT-Neo model comes in a variety of different model sizes as well ranging from 2.7 billion parameters to 125 million. For the purposes of this research, the 1.3 billion parameter model was utilized. Evaluating this model's performance in log analysis will not only further expand the range of different LLMs being evaluated, but will also enable an efficiency analysis of the model with more than 1 billion parameters (Table 5).[4]

### 3.4 LLM Sequence Classification

The sequence classification module seen in Fig. 1 shows the adaptation of an LLM for sequence classification. For sequence classification tasks using models like BERT, a dense fully connected linear layer is typically added after the final layer of the model. This then uses the output of the final transformer layer to map to a probability distribution for classification categories. This process, which employs supervised learning, requires labelled data to fine-tune the linear layer's and the language model's weights for precise classification, and adopts a loss function such as cross-entropy:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(p_{i,c}), \tag{1}$$

where:

- $N$ is the total number of samples in the dataset.
- $C$ is the number of classes in the classification problem.
- $y_{i,c}$ is a binary indicator (0 or 1) if class label $c$ is the correct classification for observation $i$.
- $p_{i,c}$ is the predicted probability that observation $i$ belongs to class $c$.

[4] All other models have just over 100 million parameters.

Cross-entropy quantifies the difference between the predicted probability distribution $p_{i,c}$ and the actual distribution $y_{i,c}$, guiding the model's training by minimizing this discrepancy. During fine-tuning, the model is initialized with a pre-trained tokenizer, equipped with a linear classification layer, and configured with specific training parameters (optimizer, learning rate, and loss function). Fine-tuning uses the model's inherent training objective masked language modelling for BERT-like models or causal language modelling for GPT-like models. Although comprehensive fine-tuning can be computationally demanding, an efficient approach is to freeze the pre-trained layers, refining only the classification layer to reduce computational demands significantly.

## 3.5 Interpretation and Visualization

The Interpretation and Visualization module of the LLM4Sec pipeline is responsible for generating visualizations that improve the explainability of the LLMs. This module accepts the vector log embedding representation as input, as well as the sequence classifier's predicted labels, and true labels in order to feed the t-SNE, and SHAP algorithms utilized in this module. The t-SNE plot is generated from the full dataset while the SHAP visualization enables log-specific explainability. An overview of this module is seen in Fig. 1.

### 3.5.1 SHAP

TranSHAP, developed by Kokalj et al. [31] extends the Kernel SHAP method to explain the decision-making process of transformer-based language models. Kernel SHAP itself is an interpretable machine learning approach, which approximates Shapley values to determine feature importance for any given model prediction. TranSHAP adapts this to transformer models by first fragmenting input sequences into indexable parts, and then creating perturbed versions of these sequences to analyze. It uses the model's tokenizer and classifier to predict outcomes of perturbed inputs and feeds these back into the Kernel SHAP algorithm, which calculates the importance of each subword fragment. Despite the computational intensity, TranSHAP can provide valuable insights into how language models weigh individual input components during classification.

### 3.5.2 t-SNE

The t-Distributed Stochastic Neighborhood Embedding (t-SNE) is a non-linear technique used for dimensionality reduction. It is particularly effective in visualizing high-dimensional datasets albeit at high computational cost [32]. Pairwise similarities are first calculated in the high-dimensional space, after which the data is mapped onto a low-dimensional space aiming to preserve these similarities. This is achieved through iterative minimization of the Kullback–Leibler divergence between the two spaces using gradient descent, optimizing the map to reveal the data's intrinsic structure in a way that is particularly informative for visual analysis.

# 4 Evaluations and Results

## 4.1 Metrics

The performance metrics adopted in this study include F1-score, Precision, and Recall. Since all the datasets have labels provided by the authors, these metrics were calculated post-training and post-testing to measure the performance of each evaluation. The metrics were derived from the confusion matrices of the training and test evaluations for each technique used in the study.

The F1-score is a traditional method for measuring the model performance on a particular dataset, it is computed using Precision and Recall. Precision gives an indication of the percentage of true positives (in class) which are relevant while Recall gives an indication of the percentage of true positives out of all the true positives available in the data. The formulae for Precision, Recall and F1-score are given in the following Eqs. 2, 3 and 4 [33].

$$P = \frac{TP}{TP + FP}, \tag{2}$$

$$R = \frac{TP}{TP + FN}, \tag{3}$$

$$F1 = \frac{2PR}{P + R}. \tag{4}$$

Weighted F1-Score is used to calculate the per class weighted mean of all F1-scores, that is to say, the average F1-score where per class proportion, S, of the dataset is considered. This is calculated with the following formulae in Eq. 5.

$$F1_{weighted} = \sum \frac{F1_{class}}{S_{class}}. \tag{5}$$

False positive rate (FPR) is used to calculate the rate at which a model classifies negatively labelled data as positive, particularly important in the case of intrusion detection because these would imply that ANOMALOUS security events were classified as NORMAL. This is calculated in the following formulae in Eq. (6)

$$FPR = \frac{FP}{FP + TN}. \tag{6}$$

Further metrics that are of importance to consider for these models are the computational cost of inference and the cost of performing fine-tuning. Specifically, application and systems logs are fast-growing with high velocity, so inference has to keep up with new events, and fine-tuning LLMs can be a costly endeavour, possibly delaying model deployment.

**Table 6** Hyperparameters employed in this research

| Hyperparameter | Value |
|---|---|
| Learning rate | $2 \times 10^{-5}$ |
| Train batch size | 8 |
| Eval batch size | 8 |
| Seed | 42 |
| Optimizer | Adam with betas = (0.9, 0.999) and epsilon $= 1 \times 10^{-8}$ |
| LR scheduler type | Linear |
| Number of epochs | 10 (BERT, RoBERTa, DistilRoBERTa, GPT-2) 3 (GPT-Neo) |

**Table 7** Benchmarking pipeline specification

| Component | Specification |
|---|---|
| CPU | Intel Xeon Silver 4214 CPU @ 2.20 GHz |
| CPU architecture | Intel x86 |
| Number of cores | 6 cores |
| RAM | 80 GB |
| Storage | 160 GB |
| GPU | NVIDIA A100 |
| GPU architecture | NVIDIA Ampere Architecture |

### 4.2 Hyperparameters

The two components of the LLM4Sec where hyperparameter tuning occurs are in the LLM module. In this section, hyperparameters are specified to improve the quality of classification and learning of log representations. The hyperparameters for the fine-tuning of LLMs with the sequence classification objective are outlined in Table 6. These represent the recommended language modelling settings for fine-tuning LLMs.

### 4.3 Efficiency Analysis: Feature Extraction and Fine-Tuning

In this research, we fine-tune each of the five chosen Language Model Models (LLMs) using sequence classification tasks on a common computing pipeline

**Table 8** Fine-tuning time for sequence classification (min)

| LLM | Type | AA | CSIC | PKDD | Thunderbird | Spirit | BGL |
|---|---|---|---|---|---|---|---|
| DistilRoBERTa | Baseline | 13.287 | 27.550 | 32.245 | 20.557 | 9.409 | 17.261 |
| | Fine-tuned | 28.871 | 73.859 | 74.059 | 46.886 | 20.158 | 39.794 |
| BERT | Baseline | 24.220 | 58.663 | 58.763 | 45.976 | 17.893 | 36.275 |
| | Fine-tuned | 56.688 | 73.859 | 141.858 | 46.886 | 42.380 | 88.245 |
| RoBERTa | Baseline | 22.855 | 57.331 | 57.387 | 36.075 | 15.651 | 30.614 |
| | Fine-tuned | 53.868 | 140.459 | 140.659 | 88.378 | 36.663 | 75.058 |
| GPT-2 | Baseline | 30.780 | 44.056 | 135.931 | 18.981 | 11.178 | 18.370 |
| | Fine-tuned | 41.936 | 134.599 | 320.879 | 63.658 | 29.881 | 51.659 |
| GPT-Neo | Baseline | 172.161 | 449.350 | 833.166 | 172.228 | 138.661 | 173.293 |
| | Fine-tuned | 229.815 | 366.570 | 862.830 | 177.735 | 134.145 | 181.980 |

**Table 9** NLP characteristics of the application log datasets

| Dataset | Word length | Word count | Character count |
|---|---|---|---|
| AA | 12.22 | 16.85 | 222.13 |
| Standard deviation | 2.87 | 1.83 | 46.07 |
| CSIC | 15.94 | 27.55 | 467.02 |
| Standard deviation | 3.50 | 0.90 | 104.13 |
| PKDD | 11.49 | 92.1 | 1124.25 |
| Standard deviation | 0.21 | 0.50 | 3.28 |

**Table 10** NLP characteristics of the system log datasets

| Dataset | Word length | Word count | Character count |
|---|---|---|---|
| Thunderbird | 8.51 | 15.48 | 146.62 |
| Standard deviation | 3.19 | 5.32 | 69.41 |
| Spirit | 7.00 | 17.40 | 137.53 |
| Standard deviation | 0.89 | 2.73 | 22.26 |
| BGL | 10.18 | 15.34 | 160.30 |
| Standard deviation | 1.62 | 8.36 | 56.48 |

(Table 7). This fine-tuning process is computationally intensive, and the training times are detailed in Table 8. Notably, the largest model, GPT-Neo, incurs significantly higher fine-tuning costs compared to the others. This is because GPT-Neo is over 10 times the size of the other models.

Across all experiments, DistilRoBERTa consistently exhibits the shortest fine-tuning times. This is because the DistilRoBERTa model is the smallest in model size relative to the other models with only 82 million parameters.

With regards to the dataset-specific fine-tuning times, we note that PKDD takes significantly longer than the other datasets due to its verbose log line length

**Table 11** Inference time for Sequence Classification LLM experiments (logs/s)

| LLM | AA | CSIC | PKDD | Thunderbird | Spirit | BGL |
|---|---|---|---|---|---|---|
| BERT | 76.400 | 77.800 | 71.300 | 76.300 | 76.000 | 77.000 |
| DistilRoBERTa | 133.900 | 135.800 | 119.600 | 135.100 | 135.700 | 135.100 |
| GPT-2 | 65.500 | 68.800 | 63.600 | 65.900 | 65.900 | 66.000 |
| GPT-Neo | 27.300 | 19.600 | 10.600 | 32.700 | 33.200 | 30.200 |
| RoBERTa | 76.700 | 80.100 | 73.100 | 76.500 | 76.500 | 76.300 |

**Table 12** Testing F1-Scores for Sequence Classification LLM with and without fine-tuning

| LLM | Type | AA | CSIC | PKDD | TB | Spirit | BGL | Average |
|---|---|---|---|---|---|---|---|---|
| DistilRoBERTa | Baseline | 0.929 | 0.696 | 0.844 | 1.000 | 0.992 | 0.985 | **0.908** |
| | Fine-tuned | 1.000 | 0.996 | **0.991** | 1.000 | 1 | 1.000 | **0.998** |
| | Change | **0.071** | **0.300** | **0.147** | 0 | **0.008** | **0.015** | **0.090** |
| RoBERTa | Baseline | 0.832 | 0.640 | 0.677 | 0.999 | 0.987 | 0.955 | 0.848 |
| | Fine-tuned | 0.999 | **0.998** | 0.984 | 1.000 | 1 | 1 | 0.997 |
| | Change | **0.167** | **0.357** | **0.307** | **0.001** | **0.013** | **0.045** | **0.148** |
| BERT | Baseline | 0.786 | 0.530 | 0.575 | 0.977 | 0.875 | 0.898 | 0.773 |
| | Fine-tuned | 1 | 0.998 | 0.966 | 1.000 | 1 | 1 | 0.994 |
| | Change | **0.214** | **0.468** | **0.391** | **0.023** | **0.125** | **0.102** | **0.220** |
| GPT-Neo | Baseline | 0.926 | 0.700 | 0.673 | 1.000 | 0.991 | 0.996 | 0.881 |
| | Fine-tuned | 0.992 | 0.994 | 0.985 | 1.000 | 0.995 | 0.958 | 0.987 |
| | Change | **0.066** | **0.294** | **0.312** | 0.000 | **0.005** | −0.038 | **0.107** |
| GPT-2 | Baseline | 0.827 | 0.238 | 0.581 | 0.975 | 0.534 | 0.898 | 0.676 |
| | Fine-tuned | 0.833 | 0.992 | 0.959 | 0.983 | 0.993 | 0.956 | 0.953 |
| | Change | **0.006** | **0.754** | **0.378** | **0.008** | **0.459** | **0.057** | **0.277** |

Higher values are better

Bold indicates either improved value for ‘change’ or best per dataset/average F1-score

see Table 9. Alternately, Spirit provides the fastest of the fine-tuning results and this is due to its comparatively smaller average log line length see Table 10.

A clear efficiency trend is seen in sequence classification inference times shown in Table 11, where GPT-Neo shows high inference times, and this highlights the advantages of the smaller DistilRoBERTa in terms of computational efficiency.

### 4.4 Results

The results, presented in Table 12, illuminate the impact of performing fine-tuning on LLMs for sequence classification with a particular application for security tasks. The data reveal testing F1-Scores for supervised sequence classification models

**Table 13** False positive rates with and without fine-tuning

| LLM | Type | AA | CSIC | PKDD | Thunderbird | Spirit | BGL |
|---|---|---|---|---|---|---|---|
| DistilRoBERTa | Baseline | 0.407 | 0.538 | 0.391 | **0.006** | 0.009 | 0.209 |
| | Fine-tuned | **0** | 0.008 | **0.023** | **0.006** | **0** | **0.002** |
| RoBERTa | Baseline | 0.857 | 0.640 | 0.835 | 0.084 | 0.007 | 0.553 |
| | Fine-tuned | **0** | 0.004 | 0.045 | **0.006** | **0** | **0** |
| BERT | Baseline | 1 | 0.861 | 1.000 | 1 | 0.000 | 1 |
| | Fine-tuned | **0** | 0.004 | 0.096 | 0.006 | **0** | **0** |
| GPT-Neo | Baseline | 0.335 | 0.192 | 0.696 | **0.006** | 0.005 | 0.041 |
| | Fine-tuned | 0.013 | 0.014 | 0.035 | **0.006** | 0.004 | 0.502 |
| GPT-2 | Baseline | 0.875 | **0.002** | 0.756 | 1 | **0** | 1 |
| | Fine-tuned | 0.005 | 0.016 | 0.121 | 0.006 | 0.005 | 0.530 |

Lower values are better

Bold indicates best false positive rate on each dataset

that have been employed across the five LLMs (Sect. 3.3), tested against six distinct datasets (Sect. 3.1).

The results highlight that fine-tuning is not only significant but also consistently observed across models. This improvement is particularly pertinent to the precision required in intrusion detection, where the F1-Scores of classification are integral to the model's loss function during fine-tuning. On the other hand, baseline performances, where the training is confined to the classification head alone, often fall short, sometimes even completely failing to classify any anomalous event. It is also apparent that the DistilRoBERTa model provides a better baseline and post-fine-tuning F1-Scores for sequence classification tasks within the security domain.

When inspecting the false positive rates in Table 13, it becomes apparent that without comprehensive fine-tuning of the underlying LLM, the ability to classify sequences accurately for intrusion detection is significantly compromised, frequently mistaking ANOMALOUS activities for NORMAL ones. It is noteworthy that models such as DistilRoBERTa, RoBERTa, and BERT not only excel but consistently deliver some of the most accurate classification results seen in the literature for intrusion detection, achieving near or absolute-perfect scores.

These findings emphasize the critical importance of fine-tuning in supervised learning for security and reaffirm the advantageous performance of BERT-style models in such high-stakes security applications.

### 4.5 Visualization

In this section, we explore the visualizations available under different LLMs. In the ideal case, the LLM-based feature sets should be both discriminatory and interpretable.

The t-SNE plots were generated for the log embedding representations produced by the baseline and fine-tuned DistilRoBERTa models and can be seen in Fig. 4. Inspection of these plots shows effective separation of NORMAL and

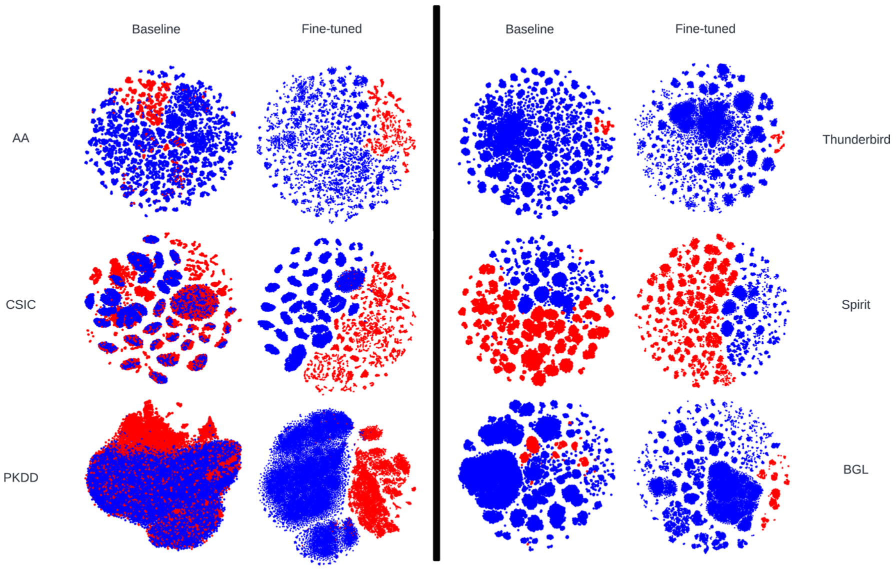
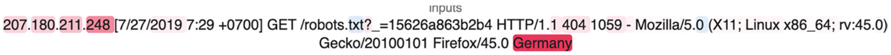


**Fig. 4** t-SNE visualizations of the embeddings produced by baseline and fine-tuned DistilRoBERTa models for all six datasets. Left (Right) pairs represent Application (System) logs

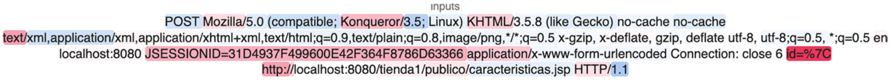


```
207.180.211.248 [7/27/2019 7:29 +0700] GET /robots.txt?_=15626a863b2b4 HTTP/1.1 404 1059 - Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0 Germany
```

**Fig. 5** SHAP explanation for log line from AA dataset using Finetuned DistilRoBERTa model

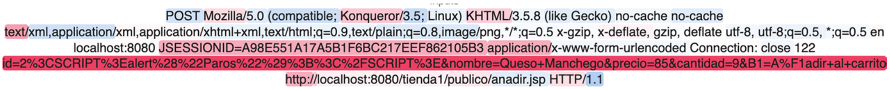
ANOMALOUS logs particularly in the system logs. While the difference between the baseline and fine-tuned models is not immediately obvious, some characteristics are shared across the visualizations which indicate some improvements in the model's ability to effectively understand the logs. Fine-tuned t-SNE plots appear to show less mixed behaviour in the clusters, and more concrete lines of separation between NORMAL, and ANOMALOUS groups. Even the PKDD dataset which is characterized by high degrees of randomness due to its anonymization sees some improvement in the disambiguation of abnormal events.

The SHAP plots were generated using unique samples from each dataset on the DistilRoBERTa model from the supervised experiments. Inspection of the different SHAP plots produced by the fine-tuned DistilRoBERTa models shows the training for sequence classification has promoted automatic feature learning in the classification models. The attention mechanisms in the LLM have been tuned to identify unique patterns present in the NORMAL and ANOMALOUS logs. Properties of notable significance include:



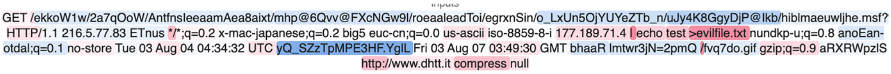
```
POST Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko) no-cache no-cache
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5 x-gzip, x-deflate, gzip, deflate utf-8, utf-8;q=0.5, *;q=0.5 en
localhost:8080 JSESSIONID=31D4937F499600E42F364F8786D63366 application/x-www-form-urlencoded Connection: close 6 id=%7C
http://localhost:8080/tienda1/publico/caracteristicas.jsp HTTP/1.1
```

**Fig. 6** SHAP explanation for first example log line from CSIC dataset using Finetuned DistilRoBERTa model



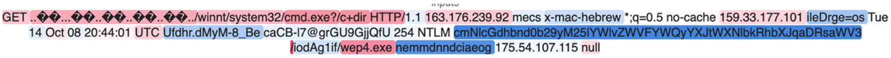
```
POST Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko) no-cache no-cache
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5 x-gzip, x-deflate, gzip, deflate utf-8, utf-8;q=0.5, *;q=0.5 en
localhost:8080 JSESSIONID=A98E551A17A5B1F6BC217EEF862105B3 application/x-www-form-urlencoded Connection: close 122
id=2%3CSCRIPT%3Ealert%28%22Paros%22%29%3B%3C%2FSCRIPT%3E&nombre=Queso+Manchego&precio=85&cantidad=9&B1=A%F1adir+al+carrito
http://localhost:8080/tienda1/publico/anadir.jsp HTTP/1.1
```

**Fig. 7** SHAP explanation for second example of log line from CSIC dataset using Finetuned DistilRoBERTa model



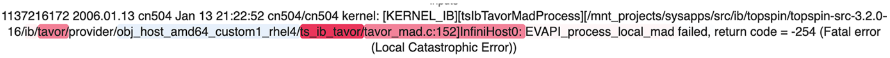
```
GET /ekkoW1w/2a7qOoW/AntfnsleeaamAea8aixt/mhp@6Qvv@FXcNGw9l/roeaaleadToi/egrxnSin/o_LxUn5OjYUYeZTb_n/uJy4K8GgyDjP@lkb/hiblmaeuwljhe.msf?
HTTP/1.1 216.5.77.83 ETnus */*;q=0.2 x-mac-japanese;q=0.2 big5 euc-cn;q=0.0 us-ascii iso-8859-8-i 177.189.71.4 | echo test >evilfile.txt nundkp-u;q=0.8 anoEan-
otdal;q=0.1 no-store Tue 03 Aug 04 04:34:32 UTC yQ_SZzTpMPE3HF.YglL Fri 03 Aug 07 03:49:30 GMT bhaaR lmtwr3jN=2pmQ /fvq7do.gif gzip;q=0.9 aRXRWpzlS
http://www.dhtt.it compress null
```

**Fig. 8** SHAP explanation for first example of log line from PKDD dataset using Finetuned DistilRoBERTa model



```
GET ..��...��..��..��..��../winnt/system32/cmd.exe?/c+dir HTTP/1.1 163.176.239.92 mecs x-mac-hebrew *;q=0.5 no-cache 159.33.177.101 ileDrge=os Tue
14 Oct 08 20:44:01 UTC Ufdhr.dMyM-8_Be caCB-l7@grGU9GjjQfU 254 NTLM cmNlcGdhbnd0b29yM25lYWlvZWVFYWQyYXJtWXNlbkRhbXJqaDRsaWV3
/iodAg1if/wep4.exe nemmdnndciaeog 175.54.107.115 null
```

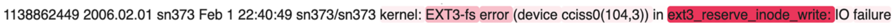
**Fig. 9** SHAP explanation for second example of log line from PKDD dataset using Finetuned DistilRoBERTa model



```
1137216172 2006.01.13 cn504 Jan 13 21:22:52 cn504/cn504 kernel: [KERNEL_IB][tsIbTavorMadProcess][/mnt_projects/sysapps/src/ib/topspin/topspin-src-3.2.0-
16/ib/tavor/provider/obj_host_amd64_custom1_rhel4/ts_ib_tavor/tavor_mad.c:152]InfiniHost0: EVAPI_process_local_mad failed, return code = -254 (Fatal error
(Local Catastrophic Error))
```

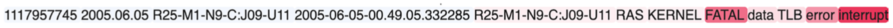
**Fig. 10** SHAP explanation for log line from Thunderbird dataset using Finetuned DistilRoBERTa model

- *AA* as seen in Fig. 5 the language model has identified the significance of individual IP addresses, location, and malformed URLs. These are features that a security expert may also evaluate in the identification of security incidents, particularly in the case of Apache web server logs.
- *CSIC* Figs. 6 and 7 illustrate how more advanced attack types are being identified by the unique characteristics of the log. The model has identified specific log characteristics associated with the use of path traversing or RCE, exploits. Even more notably one log line is flagged as anomalous for having malicious ID values submitted to the service.
- *PKDD* Figs. 8 and 9 illustrate that even though this dataset is anonymized exploits characterized by invalid file access, SQL injection, and path traversal are found in the URL and parameters have been identified by the model. Even misused credentials are being identified by the supplied parameters in this model.
- *Thunderbird* Fig. 10 shows that the model has adapted to identify log characteristics associated with kernel panic or error codes. These are learned features associated with abnormal events.



```
1138862449 2006.02.01 sn373 Feb 1 22:40:49 sn373/sn373 kernel: EXT3-fs error (device cciss0(104,3)) in ext3_reserve_inode_write: IO failure
```

**Fig. 11** SHAP explanation for log line from Spirit dataset using Finetuned DistilRoBERTa model



```
1117957745 2005.06.05 R25-M1-N9-C:J09-U11 2005-06-05-00.49.05.332285 R25-M1-N9-C:J09-U11 RAS KERNEL FATAL data TLB error interrupt
```

**Fig. 12** SHAP explanation for log line from BGL dataset using Finetuned DistilRoBERTa model

- *Spirit* Fig. 11 shows that the model has identified precise kernel error codes associated with failures similar to that of Thunderbird.
- *BGL* Fig. 12 illustrates results, in this case, the model associates the keyword 'FATAL', and 'INTERRUPT' with anomalous behaviour.

In-depth inspection of the visualizations provided by both the embeddings from the visualizations from t-SNE, and SHAP shows that the DistilRoBERTa model has captured the structure and meaning of each of the logs after fine-tuning. The pattern identification shown in the outputs of SHAP analysis indicates that the models' attention mechanisms have identified important aspects of the log line. It also shows that appropriate attention has been given to clear markers of ANOMALOUS security events like XSS, LDAP Injection, etc. This automatic feature learning results from fine-tuning the model for sequence classification. Subsequent analysis through SHAP would be of value to a security analyst since important parts of the log file which deserve further exploration are highlighted for the security analyst.

### 4.6 Discussion

In examining the performance of various LLM approaches for log analysis, especially when applying domain adaptation for sequence classification, distinct advantages emerge. Notably, the DistilRoBERTa model, upon fine-tuning, achieves near-perfect classification performance across all six datasets. This performance significantly exceeds both its baseline variant and other LLMs. Such results underscore the significance of fine-tuning, as exemplified in the CSIC dataset, where all tested LLMs experienced performance improvements ranging from 0.3 to 0.75 in terms of F1-Scores. This evidence firmly establishes the critical role of fine-tuning in optimizing LLMs for detailed and accurate log analysis tasks.

Additionally, the tranSHAP visualizations, which provide an in-depth analysis of the specific elements in a log line that identify it as an anomaly, could be highly beneficial in security analysis, particularly in cyber forensics. When examining the visualizations generated by the DistilRoBERTa model (Figs. 5, 6, 7, 10, 11, 12), the method of highlighting log characteristics is able to explicitly identify the nature of the anomalies. An excellent illustration of this is seen in Fig. 6, where the intensity of red highlighting effectively pinpoints a Cross-Site Scripting (XSS) attack as the most significant component of the log. This feature demonstrates the potential of an

**Table 14** Comparison with earlier benchmarking studies in F1-Score

| Study | CSIC | PKDD | Thunderbird | BGL | Spirit |
|---|---|---|---|---|---|
| Nguyen and Franke [5][a] | 0.9098 | 0.9256 | – | – | – |
| Bhatnagar et al. [7][a] | 0.996 | 0.996 | – | – | – |
| Wang et al. [13] | – | – | 0.99 | – | – |
| Seyyar et al. [15] | 0.96 | – | – | – | – |
| Gniewkowski et al. [20] | 0.98 | – | – | – | – |
| Farzad and Gulliver [8][a] | – | – | 0.993 | 0.994 | – |
| Guo et al. [18] | – | – | 0.998 | 0.98 | – |
| Le and Zhang [19] | – | – | 0.96 | 0.98 | 0.97 |
| Tümer Sivri et al. [9] | 0.982 | – | – | – | – |
| Adhikari and Bal [10][a] | 0.93 | – | – | – | – |
| LLM4Sec | **0.998** | 0.991 | **1.0** | **1.0** | **1.0** |

Bold indicates best F1 score

Previous results are limited to either a subset of Application or System Log files

[a]Indicates a result in which cited work reports Accuracy rather than F1-score

LLM-based approach to log analysis as a powerful tool for interpretable and accurate identification of security threats.

This study has uncovered notable benefits of using LLM-based methods for anomaly detection. One of the main advantages is the ability to produce highly interpretable results through the use of tranSHAP, coupled with a high classification performance. DistilRoBERTa, when fine-tuned for sequence classification, emerges as the most efficient model, while also offering competitive classification performance that places it among the top performing supervised methods in the field, Table 14. This approach demonstrates equal or better results compared to the existing literature in four out of five previously studied datasets, with the AA dataset being a new area of exploration. However, this high performance comes at a computational cost. Despite DistilRoBERTa nearly halving the time required for fine-tuning and inference, it remains resource-intensive, achieving a maximum classification rate of 135 logs per second and requiring over an hour for fine-tuning. While efficiency is often overlooked in the literature and challenging to benchmark due to varied experimental setups, it remains a critical factor. This is not only for ensuring swift response times to security threats in enterprise environments but also for considering the energy requirement implications of this method.

## 5 Conclusion

This research has provided a benchmark for five LLMs over six datasets for the task of log analysis in security using sequence classification. Specifically, 60 fine-tuned language models for log analysis are deployed and benchmarked. The research provides compelling results for the continued exploration of LLM-based log analysis given that the results of the sequence classification approach outperform the state-of-the-art log analysis across all datasets, especially in the case of DistilRoBERTa.

The benefits of fine-tuning can also be seen clearly in the results where it consistently provides positive improvement across all six datasets. In the case of sequence classification, the improvements range from 1 to 20% improvement. It is clear that domain adaptation of LLMs that were originally trained on text corpus can provide important improvements in the language model's ability to correctly interpret the significance of components of the log.

In addition, the opportunities for clearly legible and interpretable visualization through the t-SNE and SHAP analysis cannot be understated. The opportunity to highlight and present the components of the log line which contribute most to their classification as NORMAL or ANOMALOUS using SHAP is unique to this LLM-based approach. Such a visualization would enable security experts to see clearly why a log is considered outside of normal. Furthermore, both t-SNE and SHAP analysis visualizations show the advantages of performing fine-tuning for such analysis.

Finally, the extensive experimentation discussed throughout this research was accomplished through the use of the proposed LLM and NLP-based pipeline, LLM-4Sec. It is our aim that LLM4Sec enables research with LLM and NLP techniques in log analysis, for not only the reproducibility of the research presented in this paper but also expanding on it.

Future work will continue exploring the impact of adopting different tokenization methods for the LLMs, as well as pre-training, which historically has led to enhanced domain-specific performance. Additionally, training a new security-specific LLM from scratch with an appropriate corpus of log data using the foundational work accomplished in this study might provide much-needed efficiency improvements. Specifically, the model could afford to be much smaller with less emphasis on typical textual language corpa. Such a security-specific LLM would require an appropriate corpus of diverse log files. This would in and of itself be a novel task since there are no publicly available security logs that could provide insight into all layers of a system's architecture.

## References

1. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS), 2017, pp. 33–40 (2017). https://doi.org/10.1109/ICWS.2017.13
2. Gallagher, B., Eliassi-Rad, T.: Classification of HTTP Attacks: A Study on the ECML/PKDD 2007 Discovery challenge. Lawrence Livermore National Laboratory, Livermore (2009)
3. Boffa, M., Milan, G., Vassio, L., Drago, I., Mellia, M., Ben Houidi, Z.: Towards NLP-based processing of honeypot logs. In: 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS &PW), 2022, pp. 314–321 (2022). https://doi.org/10.1109/EuroSPW55150.2022.00038
4. Karlsen, E., Copstein, R., Luo, X., Schwartzentruber, J., Niblett, B., Johnston, A., Heywood, M.I., Zincir-Heywood, N.: Exploring semantic vs. syntactic features for unsupervised learning on application log files. In: 2023 7th Cyber Security in Networking Conference (CSNet), 2023, pp. 1–7. nPress (2023)

5. Nguyen, H.T., Franke, K.: Adaptive intrusion detection system via online machine learning. In: 2012 12th International Conference on Hybrid Intelligent Systems (HIS), 2012, pp. 271–277 (2012). https://doi.org/10.1109/HIS.2012.6421346
6. Moradi Vartouni, A., Teshnehlab, M., Sedighian Kashi, S.: Leveraging deep neural networks for anomaly-based web application firewall. IET Inf. Secur. **13**(4), 352–361 (2019)
7. Bhatnagar, M., Rozinaj, G., Yadav, P.K.: Web intrusion classification system using machine learning approaches. In: 2022 International Symposium ELMAR, 2022, pp. 57–60 (2022). https://doi.org/10.1109/ELMAR55880.2022.9899790
8. Farzad, A., Gulliver, T.A.: Log Message Anomaly Detection and Classification Using Auto-B/LSTM and Auto-GRU (2021)
9. Tümer Sivri, T., Pervan Akman, N., Berkol, A., Peker, C.: Web intrusion detection using character level machine learning approaches with upsampled data. In: Communication Papers of the 17th Conference on Computer Science and Intelligence Systems, pp. 269–274 (2022). https://doi.org/10.15439/2022F147
10. Adhikari, A., Bal, B.K.: Machine learning technique for intrusion detection in the field of the intrusion detection system (2023)
11. Copstein, R., Karlsen, E., Schwartzentruber, J., Zincir-Heywood, N., Heywood, M.: Exploring syntactical features for anomaly detection in application logs. Inf. Technol. **64**(1–2), 15–27 (2022). https://doi.org/10.1515/itit-2021-0064
12. Nam, S., Yoo, J.-H., Hong, J.W.-K.: VM failure prediction with log analysis using BERT-CNN model. In: 2022 18th International Conference on Network and Service Management (CNSM)—Mini Conference, 2022, pp. 331–337 (2022). https://doi.org/10.23919/CNSM55787.2022.9965187
13. Wang, M., Xu, L., Guo, L.: Anomaly detection of system logs based on natural language processing and deep learning. In: 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), 2018, pp. 140–144 (2018). https://doi.org/10.1109/ICFSP.2018.8552075
14. Qi, J., Luan, Z., Huang, S., Wang, Y., Fung, C., Yang, H., Qian, D.: Adanomaly: adaptive anomaly detection for system logs with adversarial learning. In: NOMS 2022—2022 IEEE/IFIP Network Operations and Management Symposium, 2022, pp. 1–5. IEEE Press (2022). https://doi.org/10.1109/NOMS54207.2022.9789917
15. Seyyar, Y.E., Yavuz, A.G., Ünver, H.M.: Detection of web attacks using the BERT model. In: 2022 30th Signal Processing and Communications Applications Conference (SIU), 2022, pp. 1–4 (2022). https://doi.org/10.1109/SIU55565.2022.9864721
16. Guo, H., Yuan, S., Wu, X.: LogBERT: log anomaly detection via BERT, pp. 1–8. IEEE (2021)
17. Shao, Y., Zhang, W., Liu, P., Huyue, R., Tang, R., Yin, Q., Li, Q.: Log anomaly detection method based on BERT model optimization. In: 2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), 2022, pp. 161–166 (2022). https://doi.org/10.1109/ICCCBDA55098.2022.9778900
18. Guo, H., Lin, X., Yang, J., Zhuang, Y., Bai, J., Zheng, T., Zhang, B., Li, Z.: TransLog: A Unified Transformer-Based Framework for Log Anomaly Detection (2022)
19. Le, V.-H., Zhang, H.: Log-Based Anomaly Detection Without Log Parsing (2021)
20. Gniewkowski, M., Maciejewski, H., Surmacz, T., Walentynowicz, W.: Sec2vec: anomaly detection in HTTP traffic and malicious URLs. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. SAC '23, 2023, pp. 1154–1162. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3555776.3577663
21. Hilmi, M.A.A., Cahyanto, K.A., Mustamiin, M.: Apache Web Server—access log pre-processing for web intrusion detection. IEEE Dataport (2020). https://doi.org/10.21227/vvvq-6w47
22. Giménez, C.T., Villegas, A.P., Maran, G.: HTTP Dataset CSIC 2010. https://doi.org/10.7910/DVN/3QBYB5
23. ECML/PKDD: ECML/PKDD 2007 Discovery Challenge (2021). https://gitlab.fing.edu.uy/gsi/web-application-attacks-datasets/-/tree/master/ecml_pkdd
24. Oliner, A.J., Stearley, J.: What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), 2007, pp. 575–584 (2007)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR (2017). arXiv:abs/1706.03762
26. Schuster, M., Nakajima, K.: Japanese and Korean voice search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 5149–5152 (2012). https://doi.org/10.1109/ICASSP.2012.6289079

27. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: a robustly optimized BERT pretraining approach. CoRR (2019). arXiv:abs/1907.11692
28. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2019). ArXiv:abs/1910.01108
29. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
30. Black, S., Leo, G., Wang, P., Leahy, C., Biderman, S.: GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata (2021). https://doi.org/10.5281/zenodo.5297715
31. Kokalj, E., Škrlj, B., Lavrač, N., Pollak, S., Robnik-Šikonja, M.: BERT meets Shapley: extending SHAP explanations to transformer-based classifiers. In: Toivonen, H., Boggia, M. (eds.) Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation, 2021, pp. 16–21. Association for Computational Linguistics (2021). https://aclanthology.org/2021.hackashop-1.3
32. Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
33. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning. The MIT Press, Cambridge (2012)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Egil Karlsen** received his Bachelor of Computer Science in 2021 with First Class Honors from Dalhousie University, where he conducted research in vulnerability analysis, authentication, and authorization services, specifically focusing on the OAuth protocol. He then earned his Master's degree in 2023, also as a member of the NIMS lab at Dalhousie, specializing in cybersecurity and machine learning. His current research focuses are in security log analysis, anomaly detection, and the application of NLP techniques in cybersecurity.

**Xiao Luo** earned her Ph.D. in computer science, specializing in natural language processing, AI, and machine learning. Presently serving as an associate professor at Oklahoma State University, one of her research areas centers on Human-XAI for Intrusion Detection Systems.

**Nur Zincir-Heywood** is a Dalhousie Distinguished Research Professor and an Associate Dean Research of Computer Science with Dalhousie University, Canada. She is the Co-Editor of the books Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning (Wiley/IEEE), and Recent Advances in Computational Intelligence in Defense and Security (Springer) as well as the Co-Author of the book Nature-Inspired Cyber Security and Resiliency: Fundamentals, Techniques, and Applications (IET). Her research interests include machine learning and artificial intelligence for cyber security, networks and systems management, and information analysis, topics on which she has published over 250 fully reviewed papers. She is a recipient of several best paper awards as well as the Supervisor for the recipient of the IFIP/IEEE IM 2013 Best Ph.D. Dissertation Award in Network Management. She is an Associate Editor of the Journal of Network and Systems Management (Springer), IEEE Transactions on Network and Service Management and the International Journal of Network Management (Wiley).

**M. Heywood** is a Professor of Computer Science at Dalhousie University (2008). He has published extensively on machine learning for over 30 years. He is an Associate Editor for IEEE Transactions on Evolutionary Computation and the Springer Journal Genetic Programming and Evolvable Machines.